Pergamon

Mathl. Comput. Modelling Vol. 26, No. 4, pp. 57–69, 1997

0895-7177/97 $17.00 + 0.00

PII: S0895-7177(97)00144-1

A Genetic Algorithm Based Multi-Dimensional Data Association Algorithm for Multi-Sensor–Multi-Target Tracking

G. CHEN AND L. HONG
Department of Electrical Engineering, Wright State University
Dayton, OH 45435, U.S.A.
lhong@cs.wright.edu

(Received and accepted June 1997)

Abstract—The central problem in multitarget-multisensor tracking is the data association problem of partitioning the observations into tracks and false alarms so that an accurate estimate of true tracks can be found. The data association problem is formed as an N-dimensional (N-D) assignment problem, which is a state-of-the-art method and is NP-hard for $N \geq 3$ sensor scans. This paper proposes a new genetic algorithm for solving the above problem which is typically encountered in the application of target tracking. The data association capacities of the genetic algorithm have been studied in different environments, and the results are presented.

Keywords—Genetic algorithms, Data association, Multisensor, Target tracking, Optimal assignment.

1. INTRODUCTION

Multidimensional assignment problems govern the central problem of data association in multisensor and multitarget tracking (MSMTT) [1,2]. Briefly, data association is the problem of partitioning observations from multiple scans of a surveillance region into tracks and false alarms. This problem of maximizing the posterior probability of the state of a surveillance region given multiple scans of the region can be formulated as a multidimensional assignment problem where the dimension refers to the number of scans of a surveillance region in a window of scans. With the recent proliferation and increasing sophistication of new technologies, system designers are recognizing that the incorporation of new techniques, such as neural networks, fuzzy logics, genetic algorithms and others, into multisensor and multitarget tracking will make a surveillance system more powerful [3–5].

In MSMTT, the key component is data association. In some situations (sparse scenarios), simple "gating" techniques might be sufficient to provide adequate data association. Medium density scenarios can be used as recursive probabilistic data association techniques. These are "soft" associations as opposed to assignment techniques, which yield "hard" associations. In dense scenarios, so-called multiple hypothesis tracking (which is a MAP (maximum *a posteriori*) technique) can be used. This is an enumerative technique which must be substantially simplified to avoid exploiting computational requirements. For a survey of the existing data association algorithms, see [6].

The main challenge in associating data from three or more scans of measurements, and the focus of this work, is that the resulting N-D (N-dimensional) assignment problem for $N \geq 3$ is NP-hard. It is well known that solving such a constrained optimization problem is intractable.

The genetic algorithm (GA) is one technique which has demonstrated significant potential as a robust and efficient method for solving an NP-hard problem [7,8]. Initially developed to analytically model the nature of genetic systems, the algorithm has experienced a widespread application to a large number of diverse optimization problems. This popularity is the result of a reputation for rapid convergence, an ability to reliably determine global optimization and, because there is no requirement for the evaluation of derivatives, the ability to deal with discontinuous costs.

In this paper, we use a new dynamic fuzzy mutation genetic algorithm to solve the multidimensional assignment problem. The paper is organized as follows. In Section 2, we formulate the multisensor multitarget measurement-to-target association problem, including real world sensor limitations such as missed detections of targets and spurious measurements, as a multidimensional assignment problem. In Section 3, we describe our proposed dynamic fuzzy mutation genetic algorithm. In Section 4, we present the simulation results of applying the proposed genetic algorithm to solve a multidimensional assignment problem for two crossing targets. Finally, some conclusions are pointed out.

2. PROBLEM FORMULATION

In this section, we formulate the data association problem for a large class of multiple target tracking as a multidimensional assignment problem. In tracking, a common surveillance problem is to estimate the past, current, or future state of a collection of objects (e.g., airplanes) moving in three-dimensional space from a sequence of measurements made of the surveillance region by one or more sensors. The objects will be called targets. The dynamics of these targets are generally modeled from physical laws of motion, but there may be noise in the dynamics and certain parameters of motion may be unknown. At time $t = 0$, a sensor (or sensors) is turned on to observe the region. In an ideal situation, measurements are taken at a finite sequence of times $\{t_k\}_{k=0}^{n}$, where $0 = t_0 < t_1 < \cdots < t_n$. At each time t_k, the sensor produces a sequence of measurements $Z(k) = \{z_{i_k}^k\}_{i_k=1}^{M_k}$, where each $z_{i_k}^k$ is a vector of noise contaminated measurements. The actual type of measurement varies with the sensor. For example, a two-dimensional radar measures the range and azimuth of each potential target $(z_{i_k}^k = (r_{i_k}^k, \theta_{i_k}^k)^\top)$. Some of the measurements may be false, and the number of targets and which measurements emanate from which target are not known.

Our goal then is to determine the number of targets, which measurements go with which targets, and which are false (i.e., the data association problem), and to estimate the state of each target given a sequence of measurements that emanate from that target. In posing the data association problem, the state of a potential target given a particular sequence of measurements from the data sets $\{Z(k)\}_{k=1}^{N}$ must be estimated. Thus, the two problems of data association and state estimation are an inseparable part of the same problem. Before proceeding to the data association problem, we briefly discuss the form of the governing equations of motion and the relation of the measurements to the state of the target.

Given the sequence of measurements $\{z(k)\}_{k=0}^{n}$ postulated to belong to a particular target, the discrete form of the dynamics of the target and the measurement equation are generally assumed to be governed by a state-space system of the form

$$\begin{aligned} x(k+1) &= f_k(x(k)) + g_k(x(k))w(k), \\ z(k) &= h_k(x(k)) + v(k), \end{aligned} \tag{1}$$

where $x(k)$ is a vector of n state variables at time t_k, $\{w(k)\}$, and $\{v(k)\}$ are independent white noise sequences of normal variables with zero mean and covariance matrices $Q(k)$ and $R(k)$, respectively, $z(k)$ represents the measurement at time k associated with this particular target, and $h_k(x(k))$ maps the state $x(k)$ to the measurement space $z(k)$.

The combinatorial optimization problem that governs a large number of data association problems in multitarget tracking and multisensor fusion is generally posed as

$$\text{maximize} \left\{ \frac{P(\Gamma = \gamma \mid Z^N)}{P(\Gamma = \gamma^0 \mid Z^N)} \;\middle|\; \gamma \subset \Gamma^* \right\}, \tag{2}$$

where Z^N represents N data sets (3), γ is a partition index of the data (and thus, induces a partition of the data), Γ^* is the finite collection of all such partitions, Γ is a discrete random element defined on Γ^*, γ^0 is a reference partition, and $P(\Gamma = \gamma \mid Z^N)$ is the posterior probability of a partition γ being true given the data Z^N.

Consider N data sets $Z(k)(k = 1, \ldots, Z(N))$ each of M_k reports $\{z_{i_k}^k\}_{i_k=1}^{M_k}$, and let Z^N denote the cumulative data set defined by

$$Z(k) = \left\{z_{i_k}^k\right\}_{i_k=1}^{M_k} \quad \text{and} \quad Z^N = \{Z(1), \ldots, Z(N)\}, \tag{3}$$

respectively.

We specialize the problem to the case of set partition defined in the following way. First, for notational convenience in representing tracks, we add a zero index to each of the index sets in (3), a dummy report z_0^k to each of the data sets $Z(k)$ in (3), and define a "track of data" as $(z_{i_1}^1, \ldots, z_{i_N}^N)$ where i_k and $z_{i_k}^k$ can now assume the values of 0 and z_0^k, respectively. A partition of the data will refer to a collection of tracks of data wherein each report occurs exactly once in one of the tracks of data and such that all data is used up; the occurrence of dummy report is unrestricted. The dummy report z_0^k serves several purposes in the representation of missing data, false reports, initiating tracks, and terminating tracks. The reference partition is one in which all reports are declared to be false. Next under appropriate independence assumptions, it can be shown that

$$\frac{P(\Gamma = \gamma \mid Z^N)}{P(\Gamma = \gamma^0 \mid Z^N)} \equiv L_\gamma \equiv \prod_{(i_1, \ldots, i_N) \subset \gamma} L_{i_1, \ldots, i_N}. \tag{4}$$

$L_{i_1 \ldots i_N}$ is a likelihood ratio containing probabilities for detection, maneuvers, and termination, as well as probability density functions for measurement errors, track initiation, and termination. Then, if $c_{i_1 \ldots i_N} = -\ln L_{i_1 \ldots i_N}$,

$$-\ln\left[\frac{P(\gamma \mid Z^N)}{P(\gamma^0 \mid Z^N)}\right] = \sum_{(i_1, \ldots, i_N) \subset \gamma} c_{i_1 \ldots i_N}. \tag{5}$$

Our goal is to find the most likely set of N-tuples such that each measurement is assigned to one and only one target, or declared false, and each target receives at most, one measurement from each scan. Using (4) and the zero-one variable $z_{i_1 \ldots i_N} = 1$ if $(i_1, \ldots, i_N) \subset \gamma$ and 0 otherwise, problem (2) can be written as the following N-dimensional assignment problem:

$$\begin{aligned}
&\text{minimize:} && \sum_{i_1=0}^{M_1} \cdots \sum_{i_N=0}^{M_N} c_{i_1, \ldots, i_N} z_{i_1 \ldots i_N}, \\
&\text{subject to:} && \sum_{i_2=0}^{M_2} \cdots \sum_{i_N=0}^{M_N} z_{i_1 \ldots i_N} = 1, \qquad i_1 = 1, \ldots, M_1, \\
& && \sum_{i_1=0}^{M_1} \cdots \sum_{i_{k-1}=0}^{M_{k-1}} \sum_{i_{k+1}=0}^{M_{k+1}} \cdots \sum_{i_N=0}^{M_N} z_{i_1 \ldots i_N} = 1, \\
& && \text{for } i_k = 1, \ldots, M_k \quad \text{and} \quad k = 2, \ldots, N-1, \\
& && \sum_{i_1=0}^{M_1} \cdots \sum_{i_{N-1}=0}^{M_{N-1}} z_{i_1 \ldots i_N} = 1, \qquad i_N = 1, \ldots, M_N, \\
& && z_{i_1 \ldots i_N} \subset \{0, 1\}, \qquad \text{for all } i_1, \ldots, i_N,
\end{aligned} \tag{6}$$

where $c_{0\ldots0}$ is arbitrarily defined to be zero. Here, each group of sums in the constraints represents the fact that each nondummy report occurs exactly once in a "track of data". One can modify this formulation to include multiassignments of one, some, or all the actual reports. The assignment problem (6) is changed accordingly. For example, if $z^k_{i_k}$ is to be assigned no more than, exactly, or no less than $n^k_{i_k}$ times, then the "$= 1$" in the constraint (6) is changed to "$\leq, =, \geq n^k_{i_k}$", respectively.

Expressions for the likelihood ratios $L_{i_1\ldots i_N}$ can be found in the work of Poore [1]. Also, they are easily modified to include target features and to account for different sensor types. In track initiation, N data sets all represent reports from N sensors, possibly all the same. For track maintenance, we use a sliding window of N data sets and one data set containing established tracks. The formulation is the same as above, except that the dimension of the assignment problem is now $N + 1$.

The complexity of this problem (6) makes its formulation and solution formidable. We have already mentioned that it is NP-hard for $N \geq 3$, even under the assumption of unity detection probability and with no spurious measurements. This is why we use the genetic algorithm to solve it.

3. GENETIC ALGORITHM SOLUTION FOR MULTIDIMENSIONAL ASSIGNMENT PROBLEM

3.1. Description of the Genetic Algorithm

The genetic algorithm is a stochastic optimization algorithm that was originally motivated by the mechanisms of natural selection and evolutionary genetics. The underlying principle of GA was first published by Holland in 1962. The mathematical framework was developed in the late 1960s, and has been presented in [7]. The GA has been proven to be efficient in many different areas, such as image processing [9], system identification [10], and fuzzy logic controller design [11]. More detail about the GA can be found in [8].

The GA is a powerful search algorithm based on the mechanics of natural selection and natural genetics. There are some differences between the GA and traditional searching algorithms. They can be summarized as follows.

- The algorithm works with a population of strings, searching many peaks in parallel, as opposed to a single point.
- The GA works directly with strings of characters representing the parameter set, not the parameters themselves.
- The GA uses probabilistic rules instead of deterministic rules.
- The GA uses objective function information instead of derivatives or other auxiliary knowledge.

The GA is inherently parallel, because it simultaneously evaluates many points in the parameter space (search space). Considering many points in the search space, the GA has a reduced chance of converging to local optimum and would be more likely to converge to global optimum. In the most conventional search techniques, a single point is considered based on some decision rules. These methods can be dangerous in multimodal (many peaks) search spaces because they can converge to local optima. However, the GA works with a population of binary strings, searching many peaks in parallel. By employing genetic operators, they exchange information between the peaks; hence, reducing the possibility of ending at a local optimum.

The GA requires only information concerning the quality of the solution produced by each parameter set (objective function values). This differs from many optimization methods which require derivative information, or, worse yet, a complete knowledge of the problem structure and parameters. Since the GA does not require such problem-specific information, it is more flexible than most search methods.

Typically, the GA is characterized by the following components.

- A genetic representation (or an encoding) for the feasible solution to the optimization problem.
- A population of encoded solution.
- A fitness function that evaluates the optimality of each solution.
- Genetic operators that generate a new population from the existing population.
- Control parameters.

The GA may be viewed as an evolutionary process wherein a population of solutions evolves into a sequence of generations. During each generation, the fitness of each solution is evaluated, and solutions are selected for reproduction based on their fitness. Selection embodies the principle of "survival of the fittest". "Good" solutions are selected for reproduction while "bad" solutions are eliminated. The "goodness" of a solution is determined from its fitness value. The selected solutions then undergo recombination under the action of the crossover and mutation operators. It has to be noted that the genetic representation may differ considerably from the natural form of the parameters of the solutions. Fixed-length and binary-coded strings for representing solutions have dominated GA research.

The power of the GA arises from crossover. Crossover causes a structured, yet randomized exchange of genetic material between solutions, with the possibility that "good" solutions can generate "better" ones.

Crossover occurs only with some probability P_c (the crossover probability or crossover rate). When the solutions are not subjected to crossover, they remain unmodified. Notable crossover techniques include the single-point, the two-points, and the uniform types [10]. Mutation involves the modification of the value of each "gene" of a solution with some probability P_m. The traditional role of mutation in GAs has been that of restoring lost or unexplored genetic material into the population to prevent the premature convergence of GAs to suboptimal solutions.

Apart from selection, crossover, and mutation, various other auxiliary operations are common in GAs. Of these, the scaling mechanism is widely used. Scaling involves a readjustment of fitness values of solutions to sustain a steady selective pressure in the population, and to prevent the premature convergence of the population to suboptimal solutions.

The basic structure of the GA is illustrated in Figure 1.

```
Simple genetic algorithm( )
{
initialize population;
evaluate population;
while convergence not achieved
        {
        scale population fitness;
        select solutions for next population;
        perform crossover and mutation;
        evaluate population;
        }
}
```

Figure 1. Basic structure of the GA.

3.2. Dynamic Fuzzy Mutation Genetic Algorithm

In the simple GA, the mutation probability P_m is a constant value. There is no uniform criteria on how to determine the appropriate P_m. Therefore, following a discussion on the search speed effect on P_m under different conditions, we can naturally introduce a fuzzy logic technique to dynamically determine P_m.

Assuming an initial string C_0 mutates to C_1, we know that the greater the probability of C_1 is better than C_0, the more beneficial it is for evolution. So for a minimum problem, we can use $P(f(C_1) < f(C_0))$ as the basis on how to select P_m. Before discussing this probability further, we first give the following two definitions.

DEFINITION 1. *The Hamming distance between strings C_1 and C_2 ($C_1, C_2 \subset IB^N = \{0,1\}^N$) is defined as*

$$D(C_1, C_2) = \sum_{i=1}^{N} |B_i(C_1) - B_i(C_2)|, \tag{7}$$

where $B_i(C)$ is the bit number $\{0,1\}$ at the i^{th} position in the string, $i = 1, 2, \ldots, N$.

DEFINITION 2. *For C_0 in problem (1), we define the following several types: $\Omega_0^{(1)}$, $\Omega_0^{(2)}, \ldots, \Omega_0^{(i)}, \ldots$, where $\Omega_0^{(i)} = \{C \mid C \subset IB^N, f(C) < f(C_0), D(C, C_0) = i\}$. Then, $\Omega_0^{(i)}$ is called i bits updated subspace of C_0; $\bigcup_{i=1,2,\ldots} \Omega_0^{(i)}$ is called the updated space of C_0, and can be written simply as $\bigcup_i \Omega_0^{(i)}$.*

It can be seen that if $C_1 \subset \bigcup_i \Omega_0^{(i)}$, then C_1 is better than C_0; otherwise, there is no improvement between C_1 and C_0. The probability of C_1 being better than C_0 equals the probability of $C_1 \subset \bigcup_i \Omega_0^i$, or

$$P(f(C_1) < f(C_0)) = P\left(C_1 \subset \bigcup_i \Omega_0^{(i)}\right). \tag{8}$$

The selection criteria of P_m are used to make the above probability as great as possible. This is the general principle in selecting P_m during the search process. In the following, we will determine the detailed expression of the above probability.

From C_0 to C_1, $D(C_0, C_1)$ bits have changed, while $N - D(C_0, C_1)$ bits have not changed, then

$$P(C_0 \to C_1) = P_m^{D(C_0,C_1)} (1 - P_m)^{(N - D(C_0,C_1))}.$$

Moreover, the following probability can be obtained:

$$P\left(C_1 \subset \Omega_0^{(i)}\right) = \left|\Omega_0^{(i)}\right| P_m^i (1 - P_m)^{(N-i)}, \tag{9}$$

where $|\Omega_0^{(i)}|$ is the element number in $\Omega_0^{(i)}$. Because $\Omega_0^{(i)} \bigcap \Omega_0^{(j)} = \emptyset$, $i \neq j$, then

$$P\left(C_1 \subset \bigcup_i \Omega_0^{(i)}\right) = \sum_i \left[\left|\Omega_0^{(i)}\right| P_m^i (1 - Pm)^{(N-i)}\right]. \tag{10}$$

THEOREM. *The probability that C_0 mutates to C_1 and enters into i bits updated subspace, denoted by $P(C_1 \subset \Omega_0^{(i)})$, takes the maximum value as $P_m = i/N$.*

PROOF. Taking a logarithm over both sides of equation (4), we have

$$\ln P\left(C_1 \subset \Omega_0^{(i)}\right) = \ln\left|\Omega_0^{(i)}\right| + i \ln P_m + (N - i) \ln(1 - Pm). \tag{11}$$

Let $\frac{\partial(\ln P(C_1 \subset \Omega_0^{(i)}))}{\partial(P_m)} = i/P_m - (N - i)/(1 - P_m = 0)$, then we have $P_m = i/N$.

In addition, $\frac{\partial^2(\ln P(C_1 \subset \Omega_0^{(i)}))}{\partial^2 P_m} = -i/P_m^2 - (N - i)/((1 - P_m)^2) < 0$, so $\ln P(C_1 \subset \Omega_0^{(i)})$ is a convex function over $P_m \subset (0,1)$, and the maximum value is achieved at saddlepoint, or $P(C_1 \subset \Omega_0^{(i)})$ takes the maximum value as $P_m = i/N$.

In many simulation experiments, we have noted the following phenomenon. During the later search period, if P_m is small, the convergence speed is very slow; if P_m is large, there are many

bits for the mutation to occur simultaneously, so it is difficult to make the search converge. Sometimes when the search falls into a local optimal point, if P_m is too small, the diversity in the population is lacking at this moment, so it is not advantageous to jump out of local optimal point and to evolve to a better search direction. We can obtain some hints on how to select the optimal P_m under above conditions via the following derivation.

1. During the later search period, every string in the generation has almost approached the optimal solution. Considering C_0 at this moment, there are not many elements belonging to its updated space; that is to say, $|\bigcup_i \Omega_0^{(i)}|$ is not great. Generally, $|\Omega_0^{(i)}| \neq 0$, then we have

$$P\left(C_1 \subset \bigcup_i \Omega_0^{(i)}\right) = \sum_i \left[\left|\Omega_0^{(i)}\right| P_m^i (1-P_m)^{N-i}\right] \tag{12}$$

$$= \left|\Omega_0^{(1)}\right| P_m (1-P_m)^{N-1} \left(1 + \frac{\left|\Omega_0^{(2)}\right|}{\left|\Omega_0^{(1)}\right|} \frac{P_m}{1-P_m} + \frac{\left|\Omega_0^{(3)}\right|}{\left|\Omega_0^{(1)}\right|} \left(\frac{P_m}{1-P_m}\right)^2 + \cdots\right).$$

   Generally, $P_m/(1-P_m) \ll 1$; thus,

$$P\left(C_1 \subset \bigcup_i \Omega_0^{(i)}\right) \leq \left|\Omega_0^{(1)}\right| P_m (1-P_m)^{N-1} \left(1 + \frac{\left|\bigcup_i \Omega_0^{(i)}\right|}{\left|\Omega_0^{(1)}\right|} \frac{P_m}{1-P_m}\right). \tag{13}$$

   It is known that if $|\bigcup_i \Omega_0^{(i)}|$ is not great, the left side of equation (13) becomes $|\Omega_0^{(1)}| P_m (1-P_m)^{(N-1)}$. This means that the probability of updated C_0 after a mutation almost equates to the probability of C_0 as it enters into 1 bit updated subspace after mutation. From the theorem, we know that optimization will be achieved as $P_m = 1/N$. This is the guideline in determining the P_m during the later period.
2. When the algorithm falls into the local optimal point (assuming C_0 arrives at local optimal), changing a few bits of C_0 cannot update the fitness value of its string. In other words, if i is not large, then $|\Omega_0^{(i)}| = 0$. At this moment, assuming that $|\Omega_0^{(r)}| \neq 0$, $|\Omega_0^{(i)}| = 0\,(i < r)$, then

$$P\left(C_1 \subset \bigcup_i \Omega_0^{(i)}\right) = \sum_{i \geq r} \left[\left[\left|\Omega_0^{(i)}\right| P_m^i\right] (1-P_m)^{(N-1)}\right]. \tag{14}$$

   From the proof of the above theorem, we know that $|\Omega_0^{(i)}| P_m^i (1-P_m)^{N-1}\,(i \geq r)$ increases progressively as $P_m \leq r/N$. Therefore, P_m is certainly greater than or equal to r/N when $P(C_1 \subset \bigcup_i \Omega_0^{(i)})$ is the maximum value. This conclusion does not give us the exact value of P_m; it only gives us the range of P_m when genetic algorithms fall into the local optimal point.

From the above analysis, it can be seen that the speed of the search requires that P_m has flexibility to some extent. In other words, P_m is a conditional probability. Of course, P_m can be expressed in a concrete functional form during the search process, such as $g(*)$, but it is difficult to determine the form of $g(*)$, so as to render a conclusion that is usually difficult to express precisely. Even though the form can render the rule of the selection of P_m, it is not easy to understand the precise function form, even to adjust and update it. To overcome the above disadvantage, we propose using fuzzy logic to characterize P_m. The main reason for using a fuzzy technique is the following. First, it is easy to render the above conclusion in the selection of P_m. Second, it is easy to embody the experience obtained from the specified problem. Third, it is easy to understand its inference form.

The implementation of a P_m selection using a fuzzy inference technique is dependent on a set of fuzzy rules, such as

$$\textit{Rule } i\textit{: IF the search enters into state } Q_i, \qquad \textit{THEN } P_m = W_i, \tag{15}$$

where Q_i is the fuzzy linguistic variable that describes the search state of GAs, W_i is the fuzzy linguistic variable that describes the value of P_m. The selection of various linguistic variables depends on the specified problem. The fuzzy rule should embody the theorem given above and the conclusion discussed, and can also include the experience obtained from the specified problem.

For example, considering a minimum problem, if P_m takes the following conditional probability:

$$P_m = P(* \mid f, df), \tag{16}$$

where f is the maximum fitness value of the population, and df is the change in f, then the fuzzy rules of P_m can be determined via the value of f and df. The rules are as the follows.

$$\begin{aligned} &\textit{Rule } 1\textit{: IF } f = PB,\ df = NB', && \textit{THEN } P_m = PM'', \\ &\textit{Rule } 2\textit{: IF } f = PB,\ df = NS', && \textit{THEN } P_m = PB'', \\ &\textit{Rule } 3\textit{: IF } f = PS, && \textit{THEN } P_m = PS'', \end{aligned} \tag{17}$$

where PB(positive big) and PS (positive small) are fuzzy linguistic variables used to characterize f, NB'(negative big) and NS''(negative small) are fuzzy linguistic variables used to characterize df, PB'' (positive big), PM'' (positive middle), and PS'' are fuzzy linguistic variables used to characterize P_m. According to the values of f and df, it can be decided which state the search will enter into; then the appropriate P_m can be selected. *Rule* 1 applies in the early period; the P_m takes the positive middle. *Rule* 2 applies when the search falls into local optimal points; P_m takes the positive big. *Rule* 3 applies in the later search period; P_m takes positive small. As for the determining of every fuzzy logic variable (including the membership functional form), it can be finished in light of the specified problem.

In the simulation experiment of this paper, the form of fuzzy rules used is similar to the one described in (17). Because f are all integer values, and the difference of f values between the two successive generations is very small (often 0 or -1) during the middle and the later search period, it is not easy to express df using NS' or NB'. Therefore, df can be defined as the difference between the maximum fitness value of the present generation and the maximum fitness value of five generations before. In addition, to simplify, $P_m = PM''$, $P_m = PB''$, and $P_m = PS''$ can be substituted by $P_m = 2/N$, $P_m = 5/N$, and $P_m = 0.8/N$, respectively. In other words, the linguistic variable membership function of P_m is taken as the form of a single point (or constant).

Heuristic knowledge can play a role in the self-adjustment of the GA mutation probability when it is implemented by a fuzzy inference system. In practice, dynamic variation of control parameters based on heuristic knowledge promises more potential to improve the efficacy of a GA search.

3.3. GA Mechanics for Multidimensional Assignment Problem

We now describe the details in employing the proposed GA to solve the multidimensional assignment problem.

3.1.1. Chromosome coding and decoding

The GA works with a population of binary strings, not the parameters themselves. For simplicity and convenience, binary coding is used in this article. With the binary coding method, the measurement number in every scan that constitute of a sequence $z_{i_1 \ldots i_N}$ would be coded as binary strings of 0's and 1's with length $B_1, \ldots, B_N$ (may be different), respectively. The choice

of $B_1, \ldots, B_N$ for the sequence is determined by the maximum measurement numbers (including the false alarm) M_i at every scan. In the binary coding method, bit length B_i and corresponding measurement number MN_i is related by

$$MN_i = \sum 2^i, \qquad i = 1, \ldots, B_i. \tag{18}$$

As a result, measurement numbers that constitutes a sequence $z_{i_1 \ldots i_N}$ can be transformed into a binary string (chromosome) with length $l_s = \sum_i B_i$. If multiple targets are considered and target number is assumed to be N_t, the string length becomes $l = l_s \times N_t$. Then the search space is explored. Note that each chromosome represents one possible solution to the problem. For example, in Table 1, each row corresponds to a track and each column to a scan. Each square in the grid stands for the event "measurement MN_i associates with track j". We assume that the conditions are as follows.

(1) There are two targets.
(2) The sensors can receive four measurements at every scan (for the reason of false alarm).
(3) The scan number in every sliding window is 5.

Then, we know that $B_i = 2$. For target one, if the corresponding sequence consists of the first measurement of scan1, the second measurement of scan2, the third measurement of scan3, the fourth measurement of scan4, the first measurement of scan5; and for target two, the corresponding sequence consists of the second measurement of scan1, the first measurement of scan2, the fourth measurement of scan3, the third measurement of scan4, the second measurement of scan 5, then the chromosome is a binary string 00 01 10 11 00 01 00 11 10 01. The decoding procedure is the reverse procedure of coding.

Table 1. Genetic algorithm code configuration for multidimensional assignment problem.

	scan1	scan2	scan3	scan4	scan5
track1	o	o	o	o	o
track2	o	o	o	o	o
track3	o	o	o	o	o
track4	o	o	o	o	o

3.3.2. Initial population

In the preparation, the second step is to form an initial population of candidate sequences. The existing sequences and the sequence that a engineer would like to start with can first be coded to form some chromosome strings in the initial population, and the rest of the initial population can be filled by randomly generated chromosomes. This ensures a direct starting point. Starting with existing knowledge in this way will usually result in a more rapid convergence, although it is possible to start from completely random chromosomes. Evolving an entire population of candidate sequences reduces the probability of landing at a local optimum.

3.3.3. Fitness and cost function

In this article, the cost function is defined as

$$E = \text{minimize} \sum_{i_1=0}^{M_1} \cdots \sum_{i_N=0}^{M_N} c_{i_1 \ldots i_N} z_{i_1 \ldots i_N}. \tag{19}$$

Our goal is to search a sequence in a constrained domain to achieve the optimization problem of (6). Hence, we need to find a suitable sequence to minimize (19). Then the cost function E takes a chromosome and returns a value. The value of the cost is then mapped into a fitness value F so as to fit into the genetic algorithm. The fitness value is a reward based on the

performance of the possible solution represented by the string. The better the solution encoded by a string, the higher the fitness. To minimize E is equivalent to getting a maximum fitness value in the genetic search process. A chromosome that has lower E should be assigned a larger fitness value. Then the genetic algorithm tries to generate better offspring to improve the fitness. Therefore, a better sequence could be obtained by a better fitness in the GA. We know that $c_{i_1 \ldots i_N}$ is always negative, so we let

$$F = \begin{cases} 0, & \text{if the constraint is violated,} \\ -E, & \text{otherwise.} \end{cases} \tag{20}$$

In evaluating the performance of a candidate sequence, the constraint on the sequence can be applied conveniently. The first part of (20) reflects a severe penalty on the fitness if the constraint is violated. Other terms that reflect explicit constraints or specifications can also be included. Searching under these constraints by the GA is much easier than minimizing a cost by a conventional optimization technique.

The simple roulette-wheel selection scheme is employed in the reproduction process for demonstration of the GA methodology in this paper. The relative fitness represents a value that is the probability of the individual in question being selected to replicate itself for generating offspring. Suppose that candidate i has fitness f_i. Then the reproduction rate is calculated by $f_i / \sum_j f_j$. For other operators, such as *crossover* and *mutation*, see [11] .

4. SIMULATION EXAMPLE

This section presents results that are designed to demonstrate the effectiveness of the proposed genetic algorithm for a multidimensional assignment problem. For this purpose we study the following problem: two targets which are moving on an $x - y$ plane with a constant velocity (Figure 2) are measured every second. The dynamic model for both targets is given by

$$X(k+1) = FX(k) + \omega(k), \qquad \omega(k) \sim N(0, Q), \tag{21}$$

where the state vector is chosen as $X = [x \dot{x} y \dot{y}]$ with the initial values $X1(0)$ =[2000 m, 15 m/s, 11000 m, $-5\,m/s$]$^{\top}$ for target one and $X2(0)$ =[2000 m, 15 m/s, 10000 m, 5 m/s]$^{\top}$ for target two, and

$$F = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
$$Q = \operatorname{diag}[0.1, 0.1, 0.1, 0.1].$$

The initial values for error covariance matrices are chosen as

$$P(0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 100000 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The correct measurements are given by

$$Z(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} X(k) + v(k), \tag{22}$$

where $v \sim N(0, \left[\begin{smallmatrix} 10000\,\mathrm{m}^2 & 0 \\ 0 & 10000\,\mathrm{m}^2 \end{smallmatrix}\right])$. The measurement data are acquired from the cluttered environment (Figure 3). The sensors are assumed with detection probability of $P_D = 1$. The

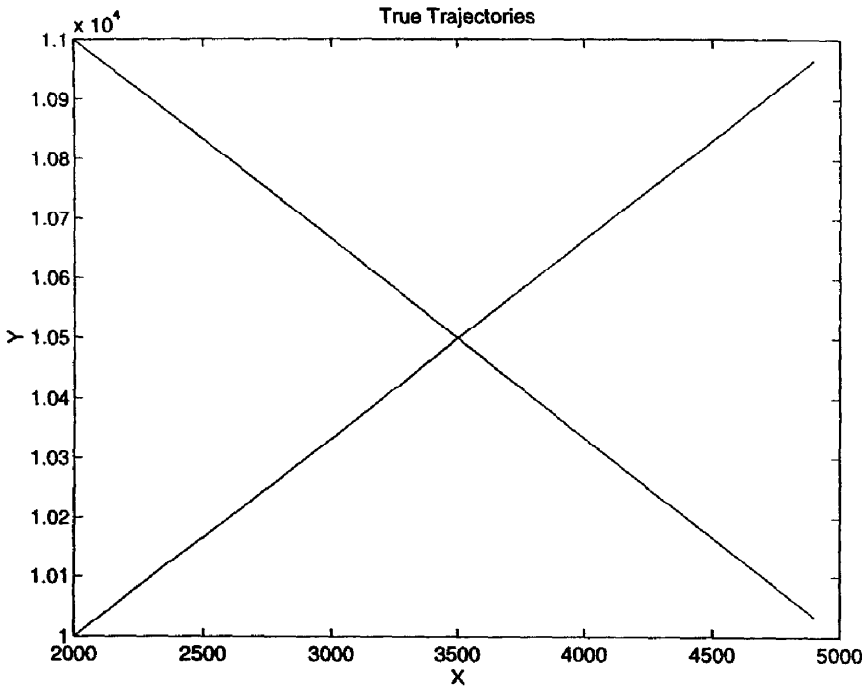

Figure 2. True target trajectories.

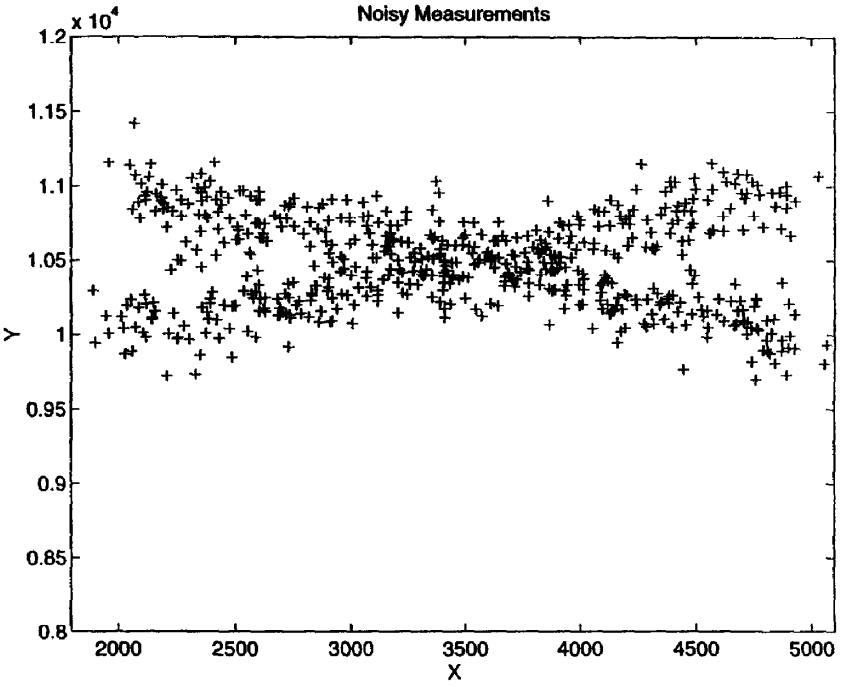

Figure 3. Noisy measurements with false alarms of the targets.

false alarm rate of the sensors is 1. With two targets, the average number of returns per scan is, therefore, four. A 10 scan window is used for tracking. During each window, there are eight new scans and two previous scans. This scenario results in a huge number of candidate associations.

In our simulations, we have used a population size of 100 for the GAs. "Scaling" of fitness values and uniform crossover probability [8] $P_c = 0.65$ have been used in the GAs. All parameters have been encoded using a fixed point encoding scheme. The mutation operator P_m is determined according to expressions given in Section 3.

In Figures 4 and 5, we present simulation results for this scenario. From the results, we know that the proposed algorithm is feasible.

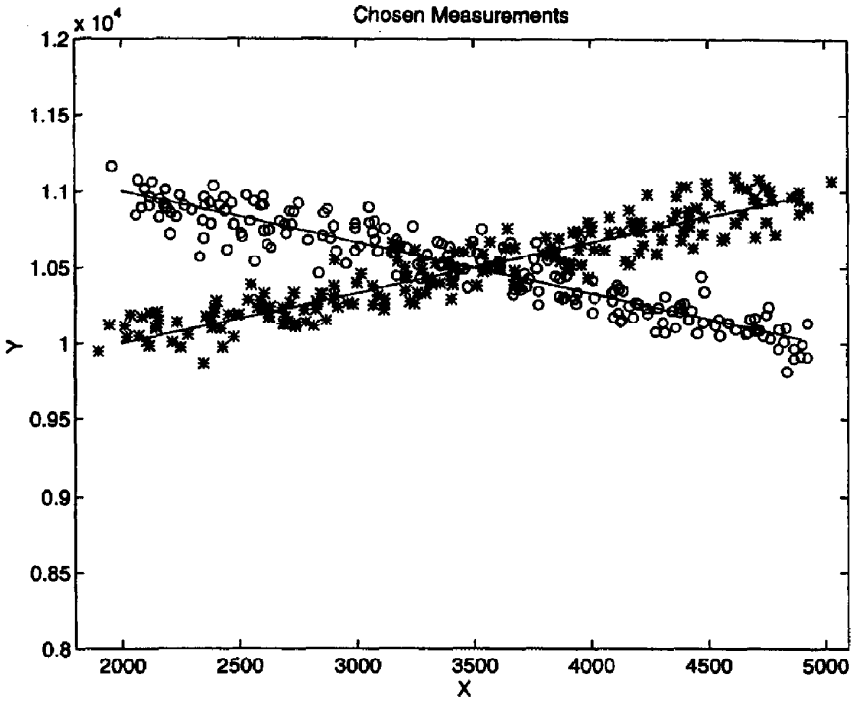

Figure 4. Selected measurements by the algorithm.

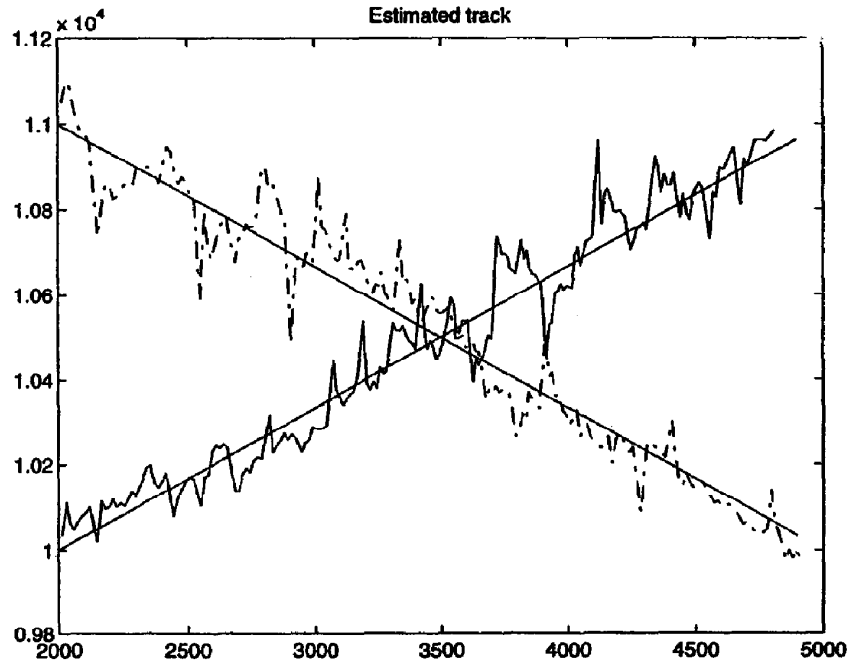

Figure 5. Estimated tracks by the algorithm.

5. CONCLUSION

In this article, we have presented a new genetic algorithm for a solution to the multidimensional assignment problem in multisensor and multitarget tracking. The algorithm works satisfactorily

on problems. As is known, track quality mainly depends on the associator's performance; therefore, the technique presented here can enhance the performance of a tracking system.

REFERENCES

1. A.B. Poore, N. Rijavec, T.N. Barker and M. Munger, Data association problems posed as multidimensional assignment problems: Numerical simulations, *SPIE* **1954** (1993).
2. S. Deb, M. Yeddanapudi, K. Pattipati and Y. Bar-shalom, A generalized S-D assignment algorithm for multisensor-multitarget state estimation, *IEEE Trans. on Aerospace and Electronic Systems* **33** (2), 523–537 (1997).
3. Y. Bar-Shalom, *Multitarget Multisensor Tracking: Advanced Applications*, Artech House, (1990).
4. F. Wang, J. Litva, T. Lo and E. Bosse, Performance of neural data associator, *IEE Proc.-Radar, Sonar Navig.* **143** (2), 71–78 (1996).
5. H. Osman, M. Farooq and T. Quach, Fuzzy logic approach to data association, *SPIE* **2755**, 313–322 (1996).
6. Y. Bar-Shalom and X. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*, YBS Publishing, Storrs, CT, (1995).
7. J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, (1975).
8. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, (1989).
9. S.K. Pal and P.P. Wang, *Genetic Algorithms for Pattern Recognition*, CRC Press, (1996).
10. L. Chambers, *Practical Handbook of Genetic Algorithms*, CRC Press, (1995).
11. G. Chen, A genetic algorithm and its applications in control engineering, Research Report of Postdoctoral Fellowship, Beijing University of Aeronautics and Astronautics, Beijing (1996).